# Estruturas de Dados - ESP412

Profª Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal de Ciência e Tecnologia da Bahia
Campus de Feira de Santana

*carolsoko@ifba.edu.br*

December 5, 2024

# Listas, Filas e Pilhas

# Listas

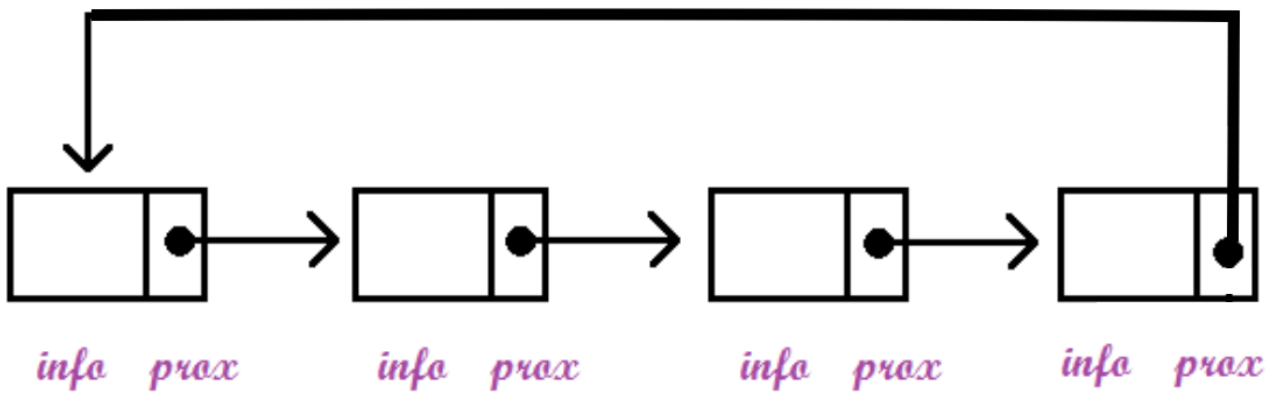
# Listas Encadeadas Circulares

# Listas Encadeadas Circulares

Uma Lista Encadeada Circular é uma estrutura de dados em que os objetos estão organizados de forma linear, assim como a Lista Simplesmente Encadeada, porém a lista não tem início, meio e fim. Ela pode iniciar de qualquer um dos seus nós. Assim, o último nó apontará para o primeiro, formando um círculo. Os nodos são encadeados de forma unidirecional, pode-se percorrer a lista em apenas um sentido, mas não na direção contrária.
[Cormen et al. 2009]

# Listas Encadeadas Circulares

Uma Lista Encadeada Circular é uma estrutura de dados em que os objetos estão organizados de forma linear, assim como a Lista Simplesmente Encadeada, porém a lista não tem início, meio e fim. Ela pode iniciar de qualquer um dos seus nós. Assim, o último nó apontará para o primeiro, formando um círculo. Os nodos são encadeados de forma unidirecional, pode-se percorrer a lista em apenas um sentido, mas não na direção contrária.
[Cormen et al. 2009]



# Listas Encadeadas Circulares

Assim como a Lista Simplesmente Encadeada, a Lista Encadeada Circular é uma lista de registros, composta de conteúdo e um ponteiro para o próximo elemento da lista. Em linguagem C, uma lista de *Structs*.

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct reg{
    int         conteudo;
    struct      reg *prox;
}celula;
```

# Listas Encadeadas Circulares

## Inserir:

O elemento pode ser inserido em qualquer posição da lista. O limite do tamanho da lista é o tamanho da memória disponível.

# Listas Encadeadas Circulares

## Inserir:

O elemento pode ser inserido em qualquer posição da lista. O limite do tamanho da lista é o tamanho da memória disponível.

```
celula *inserir(int x, celula *p){
    celula *nova, *q;
     nova = malloc (sizeof (celula));
     nova->conteudo = x;
     nova->prox = NULL;
    if (p != NULL){
        q = p;
        while(q->prox != p)
            q = q->prox;
        nova->prox = q->prox;
        q->prox = nova;
    } else {
        nova->prox = nova;
    }
    p = nova;
    return p;
}
```

# Listas Encadeadas Circulares

## Imprimir:

Exibir os elementos inseridos na lista é uma tarefa bem simples:

# Listas Encadeadas Circulares

## Imprimir:

Exibir os elementos inseridos na lista é uma tarefa bem simples:

```
void imprimir(celula *listaCircular) {
    celula *p, *topo;
    topo = listaCircular;
    for (p = listaCircular; p != NULL; p = p->prox){
        printf ("%d\n", p->conteudo);
        if (p->prox == topo) break;
    }
}
```

# Listas Encadeadas Circulares

## Buscar:

A função buscar consiste em percorrer a lista atrás da célula que contém o conteúdo específico que está sendo buscado. Assim, a função buscar deverá retornar um ponteiro para a célula da lista que contém o conteúdo de interesse:

# Listas Encadeadas Circulares

## Buscar:

A função buscar consiste em percorrer a lista atrás da célula que contém o conteúdo específico que está sendo buscado. Assim, a função buscar deverá retornar um ponteiro para a célula da lista que contém o conteúdo de interesse:

```
celula *buscar(int x, celula *listaCircular){
    celula *p, *topo;
    topo = listaCircular;
    p = listaCircular->prox;
    while (p->conteudo != x){
        p = p->prox;
        if (p == topo) break;
    }
    return p;
}
```

# Listas Encadeadas Circulares

## Remover:

A função remover pode ser construída de uma forma simples, onde passa-se o ponteiro, por valor, do elemento anterior e deseja-se remover o próximo elemento:

# Listas Encadeadas Circulares

## Remover:

A função remover pode ser construída de uma forma simples, onde passa-se o ponteiro, por valor, do elemento anterior e deseja-se remover o próximo elemento:

```
void remover(celula *p){
    celula *lixo;
    lixo = p->prox;
    p->prox = lixo->prox;
    free (lixo);
}
```

# Listas Encadeadas Circulares

## Buscar e Remover:

Ou pode-se informar o conteúdo a ser excluído. A função deverá buscar a célula correspondente e excluí-la:

# Listas Encadeadas Circulares

## Buscar e Remover:

Ou pode-se informar o conteúdo a ser excluído. A função deverá buscar a célula correspondente e excluí-la:

```
void buscar_e_remover(int y, celula *listaCircular){
    celula *p, *q, *topo;
    topo = listaCircular;
    p = listaCircular;
    q = listaCircular->prox;
    while (q->conteudo != y) {
        p = q;
        q = q->prox;
        if (q == topo) break;
    }
    if (q != topo) {
        p->prox = q->prox;
        free (q);
    }
}
```

# Listas Encadeadas Circulares

```
int main(){
    int resp;
    printf("Lista Encadeada Circular\n");
    celula *topo,*p;

    for(int j=0; j<10;j++){
        printf("Digite o próximo elemento da lista: ");
        scanf("%d", &resp);
        topo = inserir(resp, topo);
        imprimir(topo);
    }
    printf("Fim da Insercao \n");

    buscar_e_remover(5,topo);
    imprimir(topo);
    printf("removeu 5 \n");

    remover(topo->prox);
    imprimir(topo);
    printf("removeu o terceiro \n");
    return 0;
}
```

# Listas Encadeadas

## Exercício 1:

Dada uma lista encadeada informada pelo usuário, que armazena números inteiros, escreva uma função que transforma a lista em outras duas listas encadeadas: a primeira contendo os elementos cujo conteúdo é par e a segunda contendo os elementos com conteúdos ímpares. Sua função deve manipular somente os ponteiros e não o conteúdo das células.

## Exercício 2:

Dada uma lista encadeada, informada pelo usuário, que armazena números inteiros, escreva uma função que conta a quantidade de elementos desta lista. Escreva outra função que conta a quantidade de vezes que o elemento **j** se repete na lista.

# Referências

# Referências


CORMEN, T. H. et al. *Introduction to Algorithms*. 2nd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262032937.